09

Validación

Transcripción

[00:00] Ya vimos que el tema de validaciones es importante porque en este caso nuestra operación falló porque la columna 'calle' no puede estar nula. Ahora, esta validación ocurrió a nivel de la base de datos porque está especificado que es not null.

[00:17] Pero la pregunta que yo les hago a ustedes: ¿Creen que las validaciones se deberían manejar dependiendo a nivel de base de datos o ustedes creen que eso debería ser hecho unos niveles más arriba, incluso antes de que llegue a hacer contacto o interacción con la base de datos?

[00:38] Por ejemplo, aquí calle llegó nulo. La razón del problema es porque en el payload no está especificado. Entonces, en efecto, ya vimos que cuando no especificas un atributo llega nulo. Si yo aquí creo el atributo, por ejemplo que digo, "calle": "Calle 1", y envío ahora, vemos que me da un okay, entonces funcionó.

[01:05] No hay ningún error. Si le doy doble clic a mi tabla médicos, vemos aquí claramente que el id me lo autogeneró número 3. Es 3 porque yo ya hice dos pruebas antes y el id es autogenerado, entonces por esto está número 3. Y vemos claramente que los nombres que yo he enviado son los correctos, entonces ya está funcionando.

[01:29] Pero a nivel de validación si yo nuevamente le retiro algún parámetro, vamos a ver que entonces voy a tener problemas a nivel de validación. Para eso necesito usar un módulo que se llama validation. Ustedes ya saben cómo

agregar módulos, es simple, voy a detener mi servidor ¿y qué es lo primero que hago si yo necesito un módulo? Simple.

[02:10] ¿Qué es lo primero que yo hago si necesito un módulo extra? Voy a Spring Initializr, eso como yo lo dejé abierto, le voy a agregar una dependencia llamada validation, que es el bean validation con Hibernate validator. Entonces, nuevamente voy a explorar aquí y voy a buscar por la dependencia que yo acabo de agregar, que en este caso es validation.

[02:36] Esta es. Voy a copiar. Entro a mi pom.xml, voy al final y la voy a pegar aquí. Voy a guardar. Entro al módulo de Maven, servidor definido y voy a darle un refresh a mis dependencias y vemos cómo ahí comienza a resolver las dependencias del API. Como es solamente una dependencia ya terminó, muy simple, muy fácil y vamos a buscar en dónde está. Aquí al final: spring-boot-starter-validation.

[03:17] Entonces, ya estamos listos para validar a nivel del API, no de la base de datos. Ahora, si deberíamos validar donde están llegando los datos ¿dónde deberíamos hacer las validaciones? En el DTO. Entonces, llegamos aquí, DatosRegistroMedico, cerramos lo demás para no tener distracciones, close other tabs, limpiamos la terminal, un poco de orden siempre.

[03:50] Y vamos a ver que tenemos los parámetros aquí, por ejemplo, nombre, ya sabemos que no puede llegar vacío, entonces bean validation a través de anotaciones a través de anotaciones nos da facilidades, como por ejemplo si le quiero poner aquí not null va a validar que nombre nunca llegue null. Lo mismo, por ejemplo con email.

[04:15] Lo que yo voy a hacer ahora por fines de orden, voy a darles saltos de línea a cada atributo, solo para que se vea visualmente mejor y voy a iniciar las validaciones, por ejemplo, nombre puede llegar nulo o puede llegar en blanco, pero en ninguno de los casos yo debería aceptarlo.

[04:36] Entonces, recuerden, nulo no es lo mismo que vacío, entonces yo aquí le puedo poner NotBlank también. NotBlank, pero internamente NotBlank también hace lo mismo que NotNull. Entonces, voy a borrarle NotNull y esto ya va a validar que nombre no llegue ni nulo ni blanco. Es lo primero.

[05:05] Segundo el email, o sea, vamos a ponerle igual NotBlank, pero como es un email, yo quiero que valide que el formato de email es el correo. ¿Qué le pongo? Email. Perfecto. En documento vamos a hacer algo parecido, igual NotBlank porque no aceptamos ningún parámetro en blanco, pero también vamos a darle un patrón que sería una anotación pattern, porque los documentos por regla de negocio deberían tener solamente números.

[05:40] Entonces en pattern lo que hacemos aquí es definir una expresión regular en la cual vamos a ponerle backslash, dos puntos y que sea de 4 a 6 dígitos. Esa sería nuestra expresión regular que deseamos para nuestro patrón. Vemos que necesitamos el atributo regEx porque es una expresión regular como dijimos y ya está.

[06:03] Entonces eso es un número de 4 a 6 digitos, perfecto. La especialidad tampoco puede llegar null, entonces nuevamente NotBlank. Y la dirección de la misma forma, NotNull, porque es objeto. No NotBlank porque dado que es un objeto, no va a llegar en blanco, va a llegar nulo siempre, recuerden.

[06:25] Y en datos de dirección hacemos exactamente lo mismo, vamos aquí y nuevamente en calle NotBlank y esto aplicaría básicamente para todos los demás. Entonces, por fines visuales nuevamente, le voy a dar saltos de línea a cada uno. Y copiar NotBlank arriba de cada uno también. De esta forma, me aseguro que la validación ocurra a este nivel. Y estamos completos.

[06:56] Entonces voy a iniciar mi servidor nuevamente. Esperamos un poco a que mi máquina responda, ya tiene algunos años. Vemos que la aplicación inicializó y vamos a hacer la prueba nuevamente. Por ejemplo, le voy a cambiar aquí el documento, le voy a poner otro número de documento, 6

dígitos. Voy a enviarle. Vemos que dio un error aquí. ¿Y cuál es el error esta vez?

[07:29] Vamos a ver qué es lo que nos dice y dice: entrada duplicada por el email, porque esta validación sí es buena que ocurra base de datos porque el constrain, el que te va a comparar si eso ya existe o no es la llave de aquí el unique, que está aquí en la base de datos. Entonces, esa comparación sí debe ir a nivel de base de datos, porque si no, tú tendrías que hacer dos queries en una y no es el objetivo que queremos aquí.

[07:58] Entonces yo le voy a poner aquí otro identificador, un carácter2 solamente para validar que es diferente, le voy a dar enviar. Da 200. Reviso mi base de datos lista médicos y ya está con el nuevo email. ¿Ahora qué pasará? Voy a limpiar eso nuevamente. ¿Qué pasará si yo no le mando mi nombre?

[08:24] Le doy a enviar y aquí me da un error. También 500. ¿Qué me dice? La misma validación, la columna 'nombre' no puede ser nula. Vemos que es el mismo error que teníamos con la base de datos y ustedes dirán: "No, Diego, pero tú dijiste que esa validación no debía ser a nivel de base de datos. Tú dijiste que eso debía ser a nivel del objeto."

[08:49] Está bien, a nivel de objeto, pero si se dan cuenta hemos implementado las validaciones aquí, ¿pero qué falta? En el controller, en ningún momento le hemos dicho: "valídame este objeto, este DTO". ¿Por qué? Porque aquí entra la notación valid. Entonces, con valid lo que él nos dice es él va a validar que en DatosRegistroMédico todo sea válido.

[09:22] Y en DatosRegistroMédico, lo que vamos a hacer es darle un NotNull y otro valid a la dirección para que internamente también nos valide que la dirección que estamos recibiendo contenga todo lo que nosotros deseamos que contenga. Entonces voy a guardar aquí. El servidor va a reiniciar, voy a limpiar mi terminal.

[09:48] Vemos que el servidor ya inició nuevamente, y nuevamente voy a enviar aquí. Veo que tengo otro error. Pero aquí nos está diciendo: No validator could be found for NotBlank, para especialidad. ¿Por qué? Porque especialidad es un enum. Entonces nuevamente aquí también debe ser NotNull. Venimos aquí. NotNull, igual para especialidad. Guardamos, pero si se dieron cuenta, ahora ya ocurrió a nivel del bean validation y no a nivel de base de datos.

[10:24] Mis validaciones están funcionando, eso es bueno, eso es justamente lo que nosotros queremos. Limpiamos, enviamos y nos da un bad request ahora. ¿Cuál es el significado de un bad request? Que tus payloads, tus parámetros que has enviado no coinciden con el body que está esperando el API. Este tipo de respuesta es muy útil porque el cliente ya le va a entender que el error está a nivel del payload y no a nivel del servidor en sí.

[10:56] Y esto ya te ayuda mucho en la detección de errores porque no te estás preguntando: ¿será error en el servidor? ¿Será error de mi lado? No, el bad request ya te dice que el cliente está equivocado. Tú me estás enviando algo que yo no te voy a aceptar, es un bad request. ¿Y qué es lo que dice aquí? Vamos a ver.

[11:15] En springframework, en el método, etcétera, vamos a ver aquí. En el campo 'nombre': rejected value null. ¿Por qué? Porque nombre llegó en blanco. Perdón, nombre llegó null porque no está especificado. Vamos a probar ahora con nombre en blanco. Vamos aquí a nombre y que llegue en blanco. Vamos a enviar.

[11:39] Igual bad request, porque vemos aquí nuevamente dato registro en el campo nombre fue rechazado porque está vacío. De esta forma no hubo ningún tipo de interacción con la base de datos, porque la validación ocurrió a nivel del DTO. Entonces es muy importante, es muy útil porque un error 500 puede ser cualquier cosa, pero un error 400 ya te dice: "Este error no es a nivel de mi API, es a nivel del cliente".

[12:11] Esa es la parte más importante de ahora. Es prácticamente todo lo que tenemos que ver por esta clase, pero queda un último punto muy interesante porque yo me olvidé de colocar el teléfono en mi DTO. Y también en mi entidad médico. ¿Ahora qué hago? Lo vamos a ver en el siguiente video.